Python Programming - 2301CS404 Lab - 1
Het Bhalani 23010101019

01) WAP to print “Hello World”

```
print("Hello World")
```

```
Hello World
```

02) WAP to print addition of two numbers with and without using input().

```
# a,b = 1,2
# print(a+b)

a = int(input("Enter a: "))
b = int(input("Enter b: "))

print(a+b)
```

```
3
```

03) WAP to check the type of the variable.

```
a = "Het"
b = 123
c = 6.9

print(type(a),type(b),type(c))
```

```
<class 'str'> <class 'int'> <class 'float'>
```

04) WAP to calculate simple interest.

```
p = float(input("principal: "))
r = float(input("annual interest: "))
t = int(input("time: "))

i = (p * r * t) / 100

print(f"interest is: {i}")
```

05) WAP to calculate area and perimeter of a circle.

```
import math

r = float(input("Enter radius: "))
```

```
print(f"area: {math.pi*r*r}")
print(f"perimeter: {2*math.pi*r}")

area: 452.3893421169302
perimeter: 75.39822368615503
```

06) WAP to calculate area of a triangle.

```
h = int(input("enter h: "))
b = int(input("Eenter b: "))

print(f"area of triangle is {0.5*b*h}")
```

07) WAP to compute quotient and remainder.

```
a = int(input("Enter a: "))
b = int(input("Enter b: "))

print(f"quontile is {a/b} and rem is {a%b}")

quontile is 0.4 and rem is 2
```

08) WAP to convert degree into Fahrenheit and vice versa.

```
print("1. C to F")
print("2. F to C")

choice = int(input("Enter your choice (1 or 2): "))

if choice == 1:
    c = float(input("Enter temperature in Celsius: "))
    f = (c * 9 / 5) + 32
    print(f"{c}°C is equal to {f}°F.")
elif choice == 2:
    f = float(input("Enter temperature in f: "))
    c = (f - 32) * 5 / 9
    print(f"{f}°F is equal to {c}°C.")
else:
    print("Invalid choice. Please select 1 or 2.")

1. C to F
2. F to C
123.0°C is equal to 253.4°F.
```

09) WAP to find the distance between two points in 2-D space.

```
x1 = float(input("Enter x1 : "))
x2 = float(input("Enter x2 : "))
y1 = float(input("Enter y1 : "))
y2 = float(input("Enter y2 : "))
```

```
ans = ((x2 - x1)**2 + (y2-y1)**2)**(1/2)
print(ans)

1.4142135623730951
```

10) WAP to print sum of n natural numbers.

```
n = int(input("Enter a num : "))

print((n*(n+1))/2)

15.0
```

11) WAP to print sum of square of n natural numbers.

```
n = int(input("Enter a num : "))

print((n*(n+1)*(2*n+1))/6)


627874.0
```

12) WAP to concate the first and last name of the student.

```
s1 = input("Enter first name : ")
s2 = input("Enter last name : ")

print(f'{s1} {s2}')

het bhalani
```

13) WAP to swap two numbers.

```
a,b = 1,2
print(a,b)

b,a = a,b
print(a,b)

1 2
2 1
```

14) WAP to get the distance from user into kilometer, and convert it into meter, feet, inches and centimeter.

```
d = float(input("Enter distance in km : "))

print(f"{d*1000} m \n{d*3280.84} ft\n{d * 39370.1} in\n{d * 100000}
cm")
```

```
123000.0 m
403543.32 ft
4842522.3 in
12300000.0 cm
```

15) WAP to get day, month and year from the user and print the date in the given format: 23-11-2024.

```
day = int(input("Enter a date : "))
month = int(input("Enter a month : "))
year = int(input("Enter a year : "))

print(day,month,year, sep="-")
```

```
5-5-2006
```

Python Programming - 2301CS404 Lab - 2
Het Bhalani 23010101019

if..else..

01) WAP to check whether the given number is positive or negative.

```
a = int(input("Eneter a number"))

if a > 0:
    print("positive")
else:
    print("negative")
```

```
positive
```

02) WAP to check whether the given number is odd or even.

```
a = int(input("Eneter a number"))

if a%2 == 0:
    print("Even")
else:
    print("Odd")
```

```
Odd
```

03) WAP to find out largest number from given two numbers using simple if and ternary operator.

```
a = int(input("Enter a: "))
b = int(input("Enter b: "))

print("a is largest") if a>b else print("b is largest")
```

```
b is largest
```

04) WAP to find out largest number from given three numbers.

```
a = int(input("Enter a: "))
b = int(input("Enter b: "))
c = int(input("Enter c: "))

if a>b:
    if a>c:
        print("a is largest")
    else:
        print("c is largest")
```

```
else:
    if b>c:
        print("b is largest")
    else:
        print("c is largest")

c is largest
```

05) WAP to check whether the given year is leap year or not.

[If a year can be divisible by 4 but not divisible by 100 then it is leap year but if it is divisible by 400 then it is leap year]

```
y = int(input("enter a number is year"))

if y%4 == 0 and y%100!=0 or y%400 ==0:
    print("Leap year")
else:
    print("not a leap year")

Leap year
```

06) WAP in python to display the name of the day according to the number given by the user.

```
a = int(input("enter a number in 1-7"))

match a:
    case 1:
        print("Monday")
    case 2:
        print("Tuesday")
    case 3:
        print("wednesday")
    case 4:
        print("Monday")
    case 5:
        print("Monday")
    case 6:
        print("Monday")
    case 7:
        print("Monday")
```

07) WAP to implement simple calculator which performs (add,sub,mul,div) of two no. based on user input.

```
a = int(input("enter first number"))
b = int(input("enter second number"))
```

```
print("enter 1. to +\n enter 2. to -\n enter 3. to *\nenter 4. to /\
n")

c = int(input("Enter a choice"))

if(c==1):
    print(a+b)
elif(c==2):
    print(a-b)
elif(c==3):
    print(a*b)
elif(c==4):
    print(a/b)
else:
    print("Enter valid choice")
```

```
enter first number 12
enter second number 34

enter 1. to +
 enter 2. to -
 enter 3. to *
enter 4. to /


Enter a choice 1

46
```

08) WAP to read marks of five subjects. Calculate percentage and print class accordingly.

Fail below 35 Pass Class between 35 to 45 Second Class between 45 to 60 First Class between 60 to 70 Distinction if more than 70

```
l = []

for i in range(0,5):
    el = int(input())
    l.append(el)

s = (sum(l))/5

if s<35:
    print("fali")
elif s>=35 and s<45:
    print("pass")
elif s>=45 and s<60:
    print("second class")
```

```
elif s>=60 and s<70:
    print("first class")
else:
    print("distiction")


 100
 99
 98
 98
 87

distiction
```

09) Three sides of a triangle are entered through the keyboard, WAP to check whether the triangle is isosceles, equilateral, scalene or right-angled triangle.

```
a = int(input("enter first side"))
b = int(input("enter second side"))
c = int(input("enter third side"))

if a==b==c:
    print("equilateral")
elif a==b or a==c or b==c:
    print("isosceles")
elif a!=b!=c:
    print("scalene")
else:
    print("rat")

enter first side 1
enter second side 2
enter third side 3

scalene
```

10) WAP to find the second largest number among three user input numbers.

```
a = int(input("Enter a: "))
b = int(input("Enter b: "))
c = int(input("Enter c: "))

l = [a,b,c]
l.sort()

print(l[1])
```

```
Enter a:  12
Enter b:  234
Enter c:  67

67
```

11) WAP to calculate electricity bill based on following criteria. Which takes the unit from the user.

a. First 1 to 50 units – Rs. 2.60/unit

b. Next 50 to 100 units – Rs. 3.25/unit

c. Next 100 to 200 units – Rs. 5.26/unit

d. above 200 units – Rs. 8.45/unit

```python
def bilCal(units):
    bill = 0

    if units <= 50:
        bill = units * 2.60
    elif units <= 100:
        bill = (50 * 2.60) + (units - 50) * 3.25
    elif units <= 200:
        bill = (50 * 2.60) + (50 * 3.25) + (units - 100) * 5.26
    else:
        bill = (50 * 2.60) + (50 * 3.25) + (100 * 5.26) + (units - 
200) * 8.45

    return bill

units = int(input("Enter units: "))
bill = bilCal(units)

print(f"units: {units} bill: {bill:.2f}")
```

```
Enter unit:  200

19.56
```

Python Programming - 2301CS404 Lab - 3
Het Bhalani 23010101019

for and while loop

01) WAP to print 1 to 10.

```
for i in range (1,11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

02) WAP to print 1 to n.

```
n = int(input("Enter a number: "))

for i in range (0,n+1):
    print(i, end=" ")
```

```
0 1
```

03) WAP to print odd numbers between 1 to n.

```
n = int(input("Enter a number: "))

for i in range (n,n*10+1,n):
    print(i, end=" ")
```

```
5 10 15 20 25 30 35 40 45 50
```

04) WAP to print numbers between two given numbers which is divisible by 2 but not divisible by 3.

```
n = int(input("Enter a starting number: "))
m = int(input("Enter a ending number: "))


for i in range (n,m+1):
    if i%2 == 0 and i%3 != 0:
        print(i, end=" ")
```

05) WAP to print sum of 1 to n numbers.

```
n = int(input("Enter a number: "))
sum = 0

for i in range (0,n+1):
    sum += i

print(sum)
```

```
1
```

06) WAP to print sum of series 1 + 4 + 9 + 16 + 25 + 36 + ...n.

```
n = int(input("Enter a number: "))
sum = 0

for i in range (n+1):
    sum += i**2

print(sum)
```

```
1
```

07) WAP to print sum of series 1 – 2 + 3 – 4 + 5 – 6 + 7 ... n.

```
n = int(input("Enter a number: "))
sum = 0

for i in range (1,n+1):
    if i%2 != 0:
        sum+=i
    else:
        sum-=i

print(sum)
```

```
1
```

08) WAP to print multiplication table of given number.

```
n = int(input("Enter a number: "))
mul = 1

for i in range (1,11):
    print(f'{n} X {i} = {n*i}')
```

```
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
1 X 10 = 10
```

09) WAP to find factorial of the given number.

```
n = int(input("Enter a number: "))
mul = 1

for i in range (1,n+1):
    mul *= i

print(mul)
```

```
1
```

10) WAP to find factors of the given number.

```
import math

n = int(input("Enter a number: "))


for i in range (1,n+1):
    if n % i == 0:
        print(i, end=" ")
```

```
1
```

11) WAP to find whether the given number is prime or not.

```
n = int(input("Enter a number: "))
flag = False

for i in range(2,int(math.sqrt(n))+1):
    if n % i == 0:
        print("Not Prime")
        break
else:
    print("Prime")
```

```
Not a prime
```

12) WAP to print sum of digits of given number.

```
l = []
n = int(input("Enter a number: "))
sum = 0

while n>0:
    sum+=n%10
    n//=10

sum
```

```
6
```

13) WAP to check whether the given number is palindrome or not

```
n = int(input("Enter a number: "))
og_n = n
sum = 0

while n > 0:
    rem = n%10
    sum = sum*10 + rem
    n//=10

if og_n == sum:
    print("palindrom")
else:
    print("Not")
```

```
palindrom
```

14) WAP to print GCD of given two numbers.

```
a = int(input("Enter a number: "))
b = int(input("Enter a number: "))

for i in range(1, min(a,b)+1):
    if a%i ==0 and b%i == 0:
        gcd = i

gcd
```

```
4
```

Python Programming - 2301CS404 Lab - 4
Het Bhalani 23010101019

String

01) WAP to check whether the given string is palindrome or not.

```
# s = input("Enter a String: ")
# l = []
# flag = 1

# for i in s:
#     l.append(i)

# newl = l
# print(newl)
# l.reverse()
# print(l)

# for i in s:
#     if newl[i] != l[i]:
#         flag = 0
#         break

# if flag==1:
#     print("pal")
# else:
#     print("not pal")

s = input("Enter a String: ")
rev = s[::-1]

if s==rev:
    print("pal")
else:
    print("Not pal")
```

```
Not pal
```

02) WAP to reverse the words in the given string.

```
s = input("Enter a String: ")
# s[1:6:]
newl = []
l = s.split(" ")

print(s[::-1])
```

```
for i in l:
   newl.append(i[::-1])

ans = ' '.join(newl)
ans

inu rad

'rad inu'
```

03) WAP to remove ith character from given string.

```
s = input("Enter a Sting: ")
i = int(input("enter an index"))

ans = s[:i] + s[i+1:]
ans

'qwrt'
```

04) WAP to find length of string without using len function.

```
s = input("Enter a String: ")
cnt = 0

for i in s:
    cnt += 1
cnt

5
```

05) WAP to print even length word in string.

```
s = input("Enter a String: ")

l = s.split(" ")

for i in range (len(l)):
    if len(l[i]) % 2 == 0:
        print(l[i])


qw
```

06) WAP to count numbers of vowels in given string.

```
s = input("Enter a String: ")
cnt = 0

for i in range (len(s)):
    if i == 'a' or i == 'e' or i == 'i' or i == 'o' or i == 'u':
        cnt += 1

cnt
```

07) WAP to capitalize the first and last character of each word in a string.

```
def cap(s):
    words = s.split()
    result = []
    for word in words:
        if len(word) == 1:
            result.append(word.upper())
        else:
            result.append(word[0].upper() + word[1:-1] + word[-
1].upper())
    return ' '.join(result)

inputStr = "hello world python"
ans = cap(inputStr)
print(ans)

HellO WorlD PythoN

s = input("Enter a String: ")
l = s.split(" ")

for i in (l):
    print(i[0].upper() + i[1:-1] + i[-1].upper())

QwE
AsD
ZxC
```

08) WAP to convert given array to string.

```
l = ["qwer","ws",'wed']
s = ''.join(l)
s

'qwerwswed'
```

09) Check if the password and confirm password is same or not.

In case of only case's mistake, show the error message.

```
'u' == 'U'

False
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

display as : **** **** **** 3456

```
def cardNumber(cn):
    masked_number = "**** **** **** " + cn[-4:]
    return masked_number

card_number = "1234 5678 9012 3456"
newNum = cardNumber(card_number)
print(newNum)
```

```
**** **** **** 3456
```

11) : Checking if the two strings are Anagram or not.

s1 = decimal and s2 = medical are Anagram

12) : Rearrange the given string. First lowercase then uppercase alphabets.

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

```
inpt = "EHlsarwiwhtwMV"
upperStr = ""
lowerStr = ""

for i in inpt:
    if i.islower():
        lowerStr+=i
    else:
        upperStr+=i
print(lowerStr + upperStr)
```

```
lsarwiwhtwEHMV
```

List

01) WAP to find sum of all the elements in a List.

```
l = [1,2,3,4,5,6]
sum(l)
```

```
21
```

02) WAP to find largest element in a List.

```
l = [1,2,69,4,5,6]
l.sort()
l[-1]
```

```
69
```

03) WAP to find the length of a List.

```
l = [1,2,3,4,5,6]
len(l)
```

```
6
```

04) WAP to interchange first and last elements in a list.

```
l = [1,2,3,4,5,6]
l[0],l[-1] = l[-1],l[0]
l
```

```
[6, 2, 3, 4, 5, 1]
```

05) WAP to split the List into two parts and append the first part to the end.

```
l = [1,2,3,4,5,6]
l1 = l[:len(l)//2]
l2 = l[len(l)//2:]
l2.extend(l1)
l2
```

```
[4, 5, 6, 1, 2, 3]
```

06) WAP to interchange the elements on two positions entered by a user.

```
pos1 = 2
pos2 = 4
l = [1,2,3,4,5,6]

l[pos1],l[pos2] = l[pos2],l[pos1]
l

[1, 2, 5, 4, 3, 6]
```

07) WAP to reverse the list entered by user.

```
l = [1,2,3,4,5,6]
l.reverse()
l

[6, 5, 4, 3, 2, 1]
```

08) WAP to print even numbers in a list.

```
l = [1,2,3,4,5,6]

for i in range (len(l)):
    if not l[i] % 2:
       print(l[i])

2
4
6
```

09) WAP to count unique items in a list.

```
l = [1,1,3,3,5,6]
l1 = set(l)
l1 = list(l1)
cnt = 0

for i in range (len(l1)):
    cnt+=1
cnt

4
```

10) WAP to copy a list.

```
l = [1,2,3,4,5,6]
l1 = []
for i in range (len(l)):
    l1.append(l[i])
l1

[1, 2, 3, 4, 5, 6]
```

11) WAP to print all odd numbers in a given range.

```
s = 1
e = 20

for i in range (s,e+1,2):
    print(i)

1
3
5
7
9
11
13
15
17
19
```

12) WAP to count occurrences of an element in a list.

```
l = [1,2,2,4,5,6]

l.count(2)

2
```

13) WAP to find second largest number in a list.

```
l = [1,2,9,4,5,6]
l.sort()
l[-2]

6
```

14) WAP to extract elements with frequency greater than K.

```
from collections import defaultdict
l = [1,2,2,2,4,5,5,5,6,6]
ds = defaultdict(int)
k = 2

for i in l:
    ds[i] += 1

for i in ds.keys():
    if ds[i] > k:
        print(i)

2
5
```

15) WAP to create a list of squared numbers from 0 to 9 with and without using List Comprehension.

```
l = [1,2,2,4,5,6]
for i in (l):
    print(i**2)

1
4
4
16
25
36

l1 = [i**2 for i in range (0,10)]
l1

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

16) WAP to create a new list (fruit whose name starts with 'b') from the list of fruits given by user.

```
l = ['apple','banana','pineapple','branch','bhalani']
b = []

for i in l:
    if i[0] == 'b':
        b.append(i)
b

['banana', 'branch', 'bhalani']
```

17) WAP to create a list of common elements from given two lists.

```
l1 = [1,2,3,4,5]
l2 = [2,4,6]

for i in range (len(l1)):
    for j in range (len(l2)):
        if l1[i] == l2[j]:
            print(l1[i])

2
4
```

Tuple

01) WAP to find sum of tuple elements.

```
t = (1,2,3,4,5,6)
print(sum(t))
```

```
21
```

02) WAP to find Maximum and Minimum K elements in a given tuple.

```
t = (1,6,3,2,9,6)
# max,min = t[0],t[0]

# for i in t:
#     if i > max:
#         max = i
#     elif i < min:
#         min = i

# print(f'max : {max}')
# print(f'min : {min}')

k = 2
t1 = list(t)
t1.sort()

for i in range(k):
    print(t1[i])
    print(t1[-i+1])
```

```
1
2
2
1
```

03) WAP to find tuples which have all elements divisible by K from a list of tuples.

```
l = [(1,2,3,4),(5,6,7,8),(2,4,6,8),(1,2,5,7),(5,10)]
k = 2
cnt = 0

for i in l:
    for j in i:
        if j % k != 0:
            cnt += 1

    if cnt == 0:
        print(i)
    cnt = 0
```

```
(2, 4, 6, 8)
```

04) WAP to create a list of tuples from given list having number and its cube in each tuple.

```
# l = [(1,1),(2,8),(3,27),(4,64),(5,125)] ans
l = [1,2,3,4,5,6]

ans = [(i,i**3) for i in l]
ans
```

```
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125), (6, 216)]
```

05) WAP to find tuples with all positive elements from the given list of tuples.

```
l = [(1,2,3,4),(5,-6,7,8),(2,4,6,8),(1,-2,5,7),(5,-10)]
cnt = 0

for i in l:
    if all(j>0 for j in i):
        print(i)
```

```
(1, 2, 3, 4)
(2, 4, 6, 8)
```

06) WAP to add tuple to list and vice – versa.

```
t_test = (1,2,3)
l_test = [5,6,7]

og_l = [369,True,"Bhai Bhai",['a','s','d','f']]
og_l.append(t_test)

og_t = (1,2,4,6,8,4)
og_t = list(og_t)
og_t.append(l_test)
og_t = tuple(og_t)
og_t
```

```
(1, 2, 4, 6, 8, 4, [5, 6, 7])
```

07) WAP to remove tuples of length K.

```
l = [(1,2,3,6),(1,2,5,7),(5,-6,7,8,9),(2,4,6,8),(5,-10),(1,1,1,1),
(2,2,2,2)]
l2 = l.copy()
k = 4

for i in l2:
    if len(i) == k:
        l.remove(i)
print(l)
```

```
[(5, -6, 7, 8, 9), (5, -10)]
```

08) WAP to remove duplicates from tuple.

```
t = (1,1,2,3,4,3,6,8,9,9)
t = set(t)
t = tuple(t)
t
```

```
(1, 2, 3, 4, 6, 8, 9)
```

09) WAP to multiply adjacent elements of a tuple and print that resultant tuple.

```
t1 = (1,2,3,4,5)
t1 = list(t1)

ans = [(i*(i+1)) for i in range (1,len(t1))]
ans
```

```
[2, 6, 12, 20]
```

10) WAP to test if the given tuple is distinct or not.

```
l = [(1,2,3,6),(1,2,5,7),(5,-6,7,8,9),(1,2,3,6),(5,-10),(1,1,1,1),
(2,2,2,2)]

# for i in

t = (1,2,3,4,5)
t1 = list(t)
t2 = set(t)

if len(t1) == len(t2):
    print("No dups")
else:
    print("has dups")
```

```
No dups
```

```python
l = [(1,2,3,6),(1,2,5,7),(5,-6,7,8,9),(1,2,3,6),(5,-10),(1,1,1,1),
(2,2,2,2)]

for i in range(len(l)):
    for j in l[i+1:]:
        if l[i] == j:
            print(l[i])
```

```
(1, 2, 3, 6)
```

Set & Dictionary

01) WAP to iterate over a set.

```
a = {1,2,3,4,5}

for i in a:
    print(i)
```

```
1
2
3
4
5
```

02) WAP to convert set into list, string and tuple.

```
a = {1,2,3,4,5}

print(list(a))
print("".join(map(str,a)))
print(tuple(a))
```

```
[1, 2, 3, 4, 5]
12345
(1, 2, 3, 4, 5)
```

03) WAP to find Maximum and Minimum from a set.

```
b = {5,2,3,1,4}

b = list(b)
print(f'Min: {b[0]}')
print(f'Max: {b[-1]}')
```

```
Min: 1
Max: 5
```

04) WAP to perform union of two sets.

```
s1 = {1,2,3,4,5}
s2 = {6,7,8,9,10}
print(s1.union(s2))
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

05) WAP to check if two lists have at-least one element common.

```
s1 = [1,2,3,4,5]
s2 = [6,7,3,9,10]

s1 = set(s1)
s2=set(s2)

# s1 = list(s1)
# s2 = list(s2)

# for i in s1:
#     if i in s2:
#         print("True")
#         break
# else:
#     print("False")

print(s1.intersection(s2))
```

```
{3}
```

06) WAP to remove duplicates from list.

```
l = [1,2,3,5,5,3,4]
l = set(l)
l
```

```
{1, 2, 3, 4, 5}
```

07) WAP to find unique words in the given string.

```
s = "Hello maru nam het nam maru"
l = set(s.split())
l
```

```
{'Hello', 'het', 'maru', 'nam'}
```

08) WAP to remove common elements of set A & B from set A.

```
s1 = {1,2,3,4,5}
s2 = {6,7,3,9,10}

s1-s2
```

```
{1, 2, 4, 5}
```

09) WAP to check whether two given strings are anagram or not using set.

```
{3}
```

10) WAP to find common elements in three lists using set.

```python
s1 = {1,2,3,4,5}
s2 = {6,7,3,9,10}
s3 = {16,17,3,9,110}

print(s1.intersection(s2.intersection(s3)))
```

```
{3}
```

11) WAP to count number of vowels in given string using set.

```python
s = "Hello maru nam het nam maru"
```

12) WAP to check if a given string is binary string or not.

```python
s = "1010101010101000010110O1"
flag = True

for i in s:
    if (i!="0") and (i != "1"):
        flag = False

if flag:
    print("is binary")
else:
    print("Not binary")
```

```
is binary
```

13) WAP to sort dictionary by key or value.

```python
def sort_dict(d, by_key=True):
    if by_key:
        return dict(sorted(d.items()))
    else:
        return dict(sorted(d.items(), key=lambda item: item[1]))

# Example dictionary
my_dict = {'banana': 3, 'apple': 5, 'cherry': 2, 'date': 4}

# Sorting by key
sorted_by_key = sort_dict(my_dict, by_key=True)
print("Sorted by key:", sorted_by_key)

# Sorting by value
sorted_by_value = sort_dict(my_dict, by_key=False)
print("Sorted by value:", sorted_by_value)
```

14) WAP to find the sum of all items (values) in a dictionary given by user. (Assume: values are numeric)

```
def sum_of_values(d):
    return sum(d.values())

# Taking user input for dictionary
n = int(input("Enter number of items in dictionary: "))
user_dict = {}

for i in range(n):
    key = input(f"Enter key {i+1}: ")
    value = float(input(f"Enter value for {key}: "))  # Assuming
numeric values
    user_dict[key] = value

# Calculating sum of values
total = sum_of_values(user_dict)
print("Sum of all values:", total)
```

15) WAP to handle missing keys in dictionaries.

Example : Given, dict1 = {'a': 5, 'c': 8, 'e': 2}

if you look for key = 'd', the message given should be 'Key Not Found', otherwise print the value of 'd' in dict1.

```
def get_value(d, key):
    return d.get(key, "Key Not Found")

# Example dictionary
dict1 = {'a': 5, 'c': 8, 'e': 2}

# Taking user input for key
key = input("Enter the key to search: ")

# Checking for the key in dictionary
print(get_value(dict1, key))
```

User Defined Function

01) Write a function to calculate BMI given mass and height. (BMI = mass/h**2)

```python
def bmi(m,h):
    return m/(h**2)

print(bmi(45,180))
```

```
0.001388888888888889
```

02) Write a function that add first n numbers.

```python
def addn(n):
    add = 0
    for i in range(n+1):
        add+=i
    return add

print(addn(10))
```

```
55
```

03) Write a function that returns 1 if the given number is Prime or 0 otherwise.

```python
def isPrime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return 0
    return 1

print(isPrime(7))
```

```
1
```

04) Write a function that returns the list of Prime numbers between given two numbers.

```python
def isPrime(n1,n2):
    l = []
    for i in range (n1,n2+1):
        if i < 2:
            continue
        for j in range(2, int(i ** 0.5) + 1):
            if i % j == 0:
                break
        else:
            l.append(i)
    return l
print(isPrime(10,40))
```

```
[11, 13, 17, 19, 23, 29, 31, 37]
```

05) Write a function that returns True if the given string is Palindrome or False otherwise.

```
def isPalindrome(s):
    rev = s[::-1]
    if rev == s:
        return True
    else:
        return False

print(isPalindrome("helleh"))
```

```
True
```

06) Write a function that returns the sum of all the elements of the list.

```
def sumOfList(l):
    temp = sum(l)
    return temp

l = [1,2,4,6,8,9]
print(sumOfList(l))
```

```
30
```

07) Write a function to calculate the sum of the first element of each tuples inside the list.

```
def sumOfTpl(l):
    sum = 0
    for i in l:
        if i:
            sum+=i[0]
    return sum

l = [(1,2,3),(4,5),(6,7,8),(9,)]
print(sumOfTpl(l))
```

```
20
```

08) Write a recursive function to find nth term of Fibonacci Series.

```
def findNthFibb(n):
    if n<=1:
        return n

    return findNthFibb(n-1) + findNthFibb(n-2)

findNthFibb(6)
```

```
8
```

09) Write a function to get the name of the student based on the given rollno.

Example: Given dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'} find name of student whose rollno = 103

```
def findStu(roll,d):
    return d[roll]

d = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}
findStu(104,d)
```

```
'Pooja'
```

10) Write a function to get the sum of the scores ending with zero.

Example : scores = [200, 456, 300, 100, 234, 678]

Ans = 200 + 300 + 100 = 600

```
def sumOfScore(l):
    sum = 0
    for i in l:
        if i%10 == 0:
            sum += i
    return sum

l = [200, 456, 300, 100, 234, 678]
sumOfScore(l)
```

```
600
```

11) Write a function to invert a given Dictionary.

hint: keys to values & values to keys

Before : {'a': 10, 'b':20, 'c':30, 'd':40}

After : {10:'a', 20:'b', 30:'c', 40:'d'}

```
def invertDict(d1):
    d2 = {v:k for k,v in d1.items()}
    return d2

d1 = {'a': 10, 'b':20, 'c':30, 'd':40}
invertDict(d1)
```

```
{10: 'a', 20: 'b', 30: 'c', 40: 'd'}
```

12) Write a function to check whether the given string is Pangram or not.

hint: Pangram is a string containing all the characters a-z atlest once.

"the quick brown fox jumps over the lazy dog" is a Pangram string.

```
def isPan(s1):
    alphas = "abcdefghijklmnopqrstuvwxyz"
    check = True
    for i in alphas:
        if i not in s1:
            check = False
            break
    return check

s = "the quick brown fox jumps over the lazy dog"
isPan(s)
# alphas = "qwertyuiopasdfghjklzxcvbnm"
# new_str = "".join(sorted(alphas))
# print(new_str)
```

```
True
```

13) Write a function that returns the number of uppercase and lowercase letters in the given string.

example : Input : s1 = AbcDEfgh ,Ouptput : no_upper = 3, no_lower = 5

```
def findLowHigh(s1):
    l=0
    u=0
    for i in s1:
        if i.islower():
            l+=1
        elif i.isupper():
            u+=1
    l = [l,u]
    return l

findLowHigh("AbcDEfgh")
```

```
[5, 3]
```

14) Write a lambda function to get smallest number from the given two numbers.

```
x = lambda a,b : max(a,b)
print(x(10,20))
```

```
20
```

15) For the given list of names of students, extract the names having more that 7 characters. Use filter().

```
students = ["Jonathan", "Alice", "Christopher", "David", "Elizabeth",
"John", "Catherine"]

long_names = list(filter(lambda name: len(name) > 7, students))

print("Names with more than 7 characters:", long_names)
```

16) For the given list of names of students, convert the first letter of all the names into uppercase. use map().

```
students = ["jonathan", "alice", "christopher", "david", "elizabeth",
"john", "catherine"]

capitalized_names = list(map(lambda name: name.capitalize(),
students))

print("Names with first letter capitalized:", capitalized_names)
```

```
Names with first letter capitalized: ['Jonathan', 'Alice',
'Christopher', 'David', 'Elizabeth', 'John', 'Catherine']
```

17) Write udfs to call the functions with following types of arguments:

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwargs)

5. Keyword-Only & Positional Only Arguments

```python
def positional_args(name, age):
    print(f"Name: {name}, Age: {age}")

def keyword_args(name, age):
    print(f"Name: {name}, Age: {age}")

def default_args(name, age=18):
    print(f"Name: {name}, Age: {age}")

def variable_length_args(*args, **kwargs):
    print("Positional Arguments:", args)
    print("Keyword Arguments:", kwargs)

def mixed_arguments(a, /, b, *, c):
    print(f"Positional-Only: {a}, Regular: {b}, Keyword-Only: {c}")

# Calling functions
print("1. Positional Arguments:")
positional_args("Alice", 25)

print("\n2. Keyword Arguments:")
keyword_args(age=30, name="Bob")

print("\n3. Default Arguments:")
default_args("Charlie")  # Uses default age = 18
default_args("Charlie", 22)  # Overrides default

print("\n4. Variable Length Arguments:")
variable_length_args(1, 2, 3, name="David", age=40)

print("\n5. Keyword-Only & Positional-Only Arguments:")
mixed_arguments(10, b=20, c=30)
```

File I/O

01) WAP to read and display the contents of a text file. (also try to open the file in some other directory)

- in the form of a string

- line by line

- in the form of a list

```
fp = open('a.txt','r')
# print(fp.read())
# print(fp.readline())
print(fp.readlines())
fp.close()
```

```
['qwertyuiop asdfghjkl zxcvbnm\n', 'mnbvcxz lkjhgfdsa\n',
'poiuytrewq']
```

02) WAP to create file named "new.txt" only if it doesn't exist.

```
fp = open('new.txt','w')
fp.close()
```

03) WAP to read first 5 lines from the text file.

```
fp = open('a.txt','r')

for i in range(5):
    print(fp.readline())
fp.close()
```

```
qwertyuiop asdfghjkl zxcvbnm

mnbvcxz lkjhgfdsa

poiuytrewq

qwertyuiop asdfghjkl zxcvbnm

mnbvcxz lkjhgfdsa
```

04) WAP to find the longest word(s) in a file

```
fp1 = open('a.txt','r')
l = fp1.read().split()

max = 0

for i in l:
    if len(i) > max:
        max = len(i)

for i in l:
    if len(i) == max:
        l1.append(i)

print(l1)
l1.clear()
```

```
['brooooooo']
```

05) WAP to count the no. of lines, words and characters in a given text file.

06) WAP to copy the content of a file to the another file.

```
fp1 = open('a.txt','r')
l = fp1.read()
fp1.close()

fp2 = open('b.txt','w')
fp2.write(l)
fp2.close()
```

07) WAP to find the size of the text file.

```
fp1 = open('a.txt','r')
print(f'chars:{len(fp1.read()) * 50}')
```

```
chars:1900
```

08) WAP to create an UDF named frequency to count occurances of the specific word in a given text file.

```
def frq(word):
    fp1 = open('a.txt','r')
    l = fp1.read().split()
    cnt = 0

    for i in l:
        if i == word:
            cnt+=1

    return cnt

print(frq('bro'))
```

```
2
```

09) WAP to get the score of five subjects from the user, store them in a file. Fetch those marks and find the highest score.

```
l = [10,52,48,97,63]

fp1 = open('marks.txt','w')
l2 = [fp1.write(str(i)+" ") for i in l]
fp1.close()

fp2 = open('marks.txt','r')
l3 = fp2.read().split()
l4 = [int(i) for i in l3]
l4.sort()
l4[-1]
```

```
97
```

10) WAP to write first 100 prime numbers to a file named primenumbers.txt

(Note: each number should be in new line)

```
fp = open('prime.txt','w+')
import math
def isPrime(n):
    for i in range(2,int(math.sqrt(n))+1):
        if n % i == 0:
            break
    else:
        return(n)

l = []
i = 2
while(len(l) != 100):
    if isPrime(i):
        l.append(i)
    i+=1

for i in l:
    fp.write(str(i)+'\n')
fp.close()
```

11) WAP to merge two files and write it in a new file.

```
def merge_files(file1, file2, output_file):
    try:
        with open(file1, 'r') as f1, open(file2, 'r') as f2,
open(output_file, 'w') as out:
            out.write(f1.read())
            out.write("\n")
            out.write(f2.read())
        print(f"Files merged successfully into '{output_file}'")
    except FileNotFoundError as e:
        print(f"Error: {e}")

file1 = "file1.txt"
file2 = "file2.txt"
output_file = "merged.txt"

merge_files(file1, file2, output_file)
```

12) WAP to replace word1 by word2 of a text file. Write the updated data to new file.

```
def replace_word(input_file, word1, word2, output_file):
    try:
        with open(input_file, 'r') as file:
            data = file.read()

        updated_data = data.replace(word1, word2)

        with open(output_file, 'w') as file:
            file.write(updated_data)

        print(f"Replaced '{word1}' with '{word2}' and saved in
'{output_file}'")

    except FileNotFoundError as e:
        print(f"Error: {e}")

input_file = "input.txt"
output_file = "output.txt"
word1 = "oldword"
word2 = "newword"

replace_word(input_file, word1, word2, output_file)
```

13) Demonstrate tell() and seek() for all the cases(seek from beginning-end-current position) taking a suitable example of your choice.

```python
with open("example.txt", "w") as file:
    file.write("Hello, this is a sample text.")

with open("example.txt", "r") as file:
    print("Initial Position:", file.tell())

    file.seek(7)
    print("After seeking to position 7:", file.tell(), "->",
file.read(5))

    file.seek(0, 1)
    print("After seeking from current position (no move):",
file.tell())

    file.seek(-5, 2)
    print("After seeking 5 bytes before end:", file.tell(), "->",
file.read())
```

Exception Handling

01) WAP to handle following exceptions:

1. ZeroDivisionError
2. ValueError
3. TypeError

   Note: handle them using separate except blocks and also using single except block too.

```
try:
    print(369/0)
except ZeroDivisionError:
    print("zero div error")

try:
    n = 'batman'
    print(100/int(n))
except ValueError:
    print('Value Error')

try:
    lol = "lol"
    num = 2
    print(lol + num + lol)
except TypeError:
    print('Type Error')
```

```
zero div error
Value Error
Type Error
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

```
try:
    a = [1,2,3,4]
    print(a[5])
except IndexError:
    print('Index error')
```

```
try:
    a = {'name' : 'lol'}
    print(a['lol'])
except KeyError:
    print('Key error')
```

```
Index error
Key error
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

```
try:
    fp = open('lol.txt','r')
except FileNotFoundError:
    print('FIle not found')
try:
    import ghar_nu_module
except ModuleNotFoundError:
    print('Moule not found')
```

```
FIle not found
Moule not found
```

04) WAP that catches all type of exceptions in a single except block.

```
try:
    print(369/0)
except:
    print('something went Wrong !')
```

```
something went Wrong !
```

05) WAP to demonstrate else and finally block.

```
try:
    print(100/0)
except ZeroDivisionError:
    print('Zero devision')
else:
    print('NO error accures')
finally:
    print('This is last Statement')
```

```
Zero devision
This is last Statement
```

06) Create a short program that prompts the user for a list of grades separated by commas.

Split the string into individual grades and use a list comprehension to convert each string to an integer.

You should use a try statement to inform the user when the values they entered cannot be converted.

```
try:
    a = input('Enter grades').split(',')
    a2 = [int(i) for i in a]
except ValueError:
    print('Grade cannot converted into int')
else:
    print('You enterd right numbers')
```

07) WAP to create an udf divide(a,b) that handles ZeroDivisionError.

```
def divide(a,b):
    try:
        print(a/b)
    except ZeroDivisionError:
        print("Can not divede by zero")

divide(10,0)
```

```
Can not divede by zero
```

08) WAP that gets an age of a person form the user and raises ValueError with error message: "Enter Valid Age" :

If the age is less than 18.

otherwise print the age.

```
class BalakHoTum(Exception):
   def __init__(self,msg):
        self.msg = msg

try:
    age = int(input("Enter a Age: "))

    if age < 18:
        raise BalakHoTum("You are underage")
    else:
        print(age)
```

```
except BalakHoTum as er:
    print(er)
```

```
You are underage
```

09) WAP to raise your custom Exception named InvalidUsernameError with the error message : "Username must be between 5 and 15 characters long":

if the given name is having characters less than 5 or greater than 15.

otherwise print the given username.

```
class InvalidUsernameError (Exception):
    def __init__(self,msg):
        self.msg = msg

try:
    s = input("Enter a Username")

    if len(s)<5 or len(s)>15:
        raise InvalidUsernameError("Bhai naam barobar nakh")
    else:
        print(s)

except InvalidUsernameError as e:
    print(e)
```

```
Bhai naam barobar nakh
```

10) WAP to raise your custom Exception named NegativeNumberError with the error message : "Cannot calculate the square root of a negative number" :

if the given number is negative.

otherwise print the square root of the given number.

```
class NegativeNumberError(Exception):
    def __init__(self,msg):
        self.msg = msg

try:
    n = int(input("Enter a number: "))

    if n < 0:
        raise NegativeNumberError("Kalpanik sankhya aavse vala")
    else:
        print(n ** (1/2))

except NegativeNumberError as e:
    print(e)
```

```
3.4641016151377544
```

Python Programming - 2301CS404 Lab - 11
Het Bhalani 23010101019

Modules

01) WAP to create Calculator module which defines functions like add, sub,mul and div.

Create another .py file that uses the functions available in Calculator module.

```
from calcModule import add, sub, mul, div

print(add(10, 20))
print(sub(10, 20))
print(mul(10, 20))
print(div(10, 20))
```

```
30
-10
200
0.5
```

02) WAP to pick a random character from a given String.

```
import random as r

s = "blablabla"

print(r.choice(s))
```

```
a
```

03) WAP to pick a random element from a given list.

```
l = [1, 2, 3, 4, 5]
print(r.choice(l))
```

```
3
```

04) WAP to roll a dice in such a way that every time you get the same number.

```
l = [1, 2, 3, 4, 5, 6]
r.seed(369)
print(r.choice(l))
```

```
3
```

05) WAP to generate 3 random integers between 100 and 999 which is divisible by 5.

```
l = [i for i in range(100,1000)]
threeRandoms = r.sample(l, 3)
threeRandoms

[895, 419, 833]
```

06) WAP to generate 100 random lottery tickets and pick two lucky tickets from it and announce them as Winner and Runner up respectively.

```
hundredRandoms = [r.randint(1, 100) for i in range(100)]
two = r.sample(hundredRandoms, 2)
r.seed(369)
winner = two[0]
runnerUp = two[1]
print(f'winner: {winner}, runnerUp: {runnerUp}')

winner: 54, runnerUp: 8
```

07) WAP to print current date and time in Python.

```
import datetime as dt

curr = dt.datetime.now()
curr

datetime.datetime(2025, 2, 18, 10, 33, 35, 584440)
```

08) Subtract a week (7 days) from a given date in Python.

```
d = dt.datetime(2025,1,17)
subWeek = d - dt.timedelta(days=7)
subWeek.day

10
```

09) WAP to Calculate number of days between two given dates.

```
d1 = dt.datetime(2025,1,17)
d2 = dt.datetime(2025,1,1)
diff = d1 - d2
diff.days

16
```

10) WAP to Find the day of the week of a given date.(i.e. wether it is sunday/monday/tuesday/etc.)

```
# d1 = dt.datetime(2025,1,17)
d1 = dt.datetime.today()
curr = d1.strftime('%a')
curr
```

```
'Tue'
```

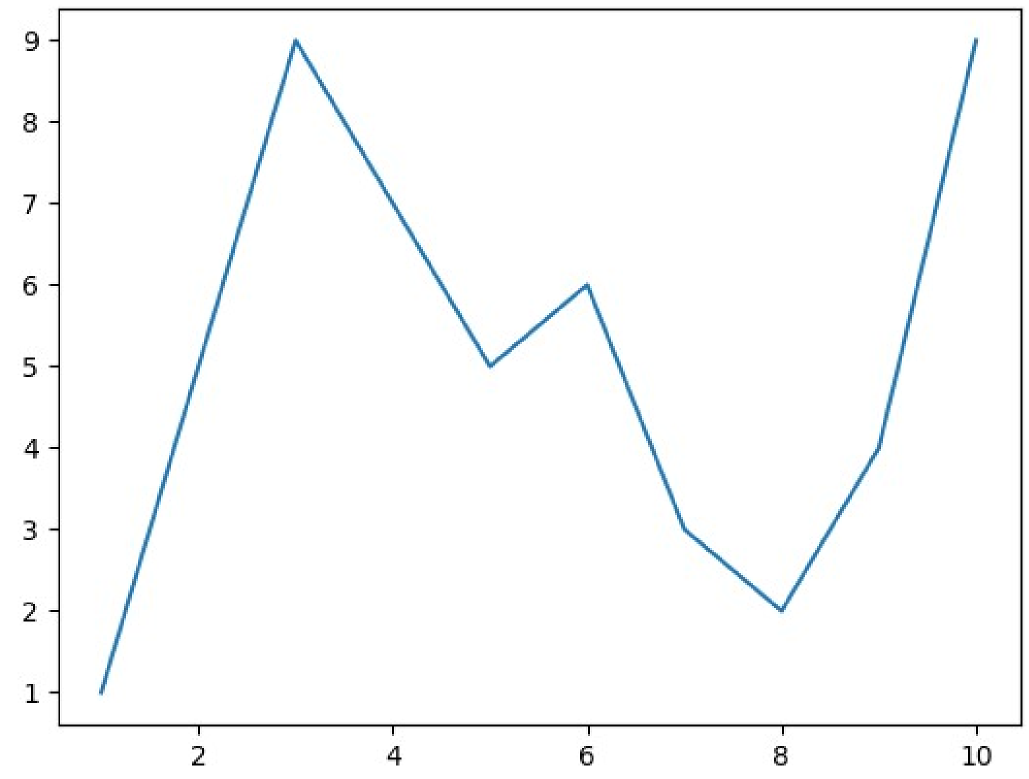
11) WAP to demonstrate the use of date time module.

```
d1 = dt.datetime.today()
# curr = d1.strftime('%a')
# curr
print(d1.strftime('%a'))
```

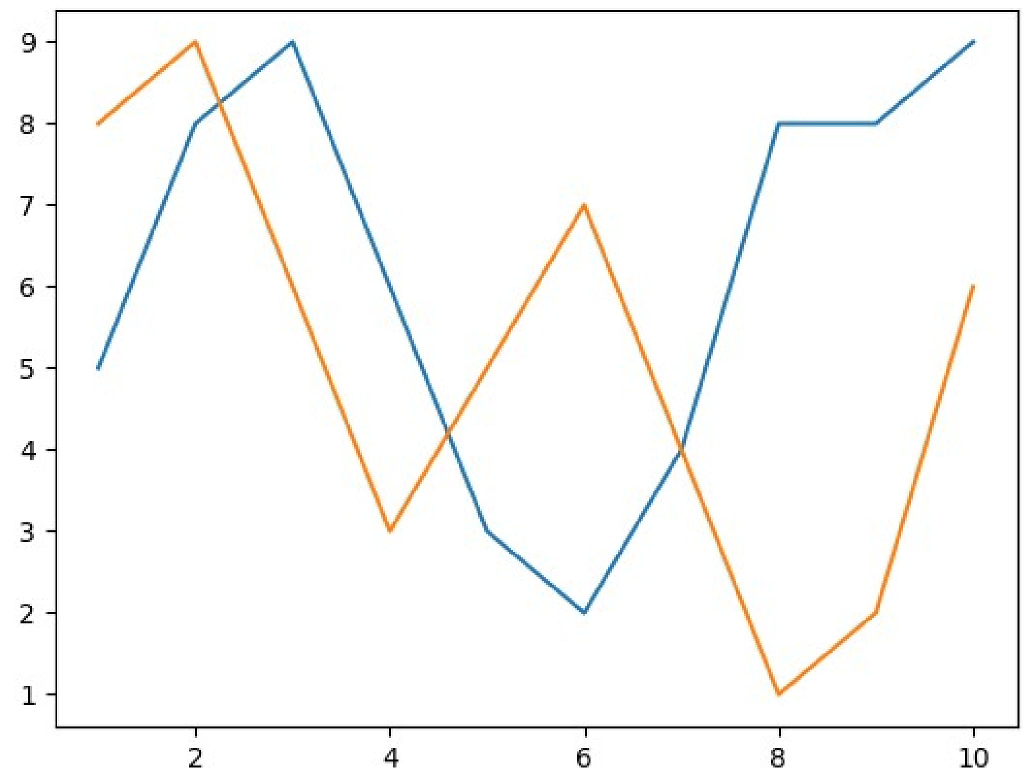
```
datetime.datetime(2025, 2, 18, 10, 35, 59, 102377)
```

12) WAP to demonstrate the use of the math module.

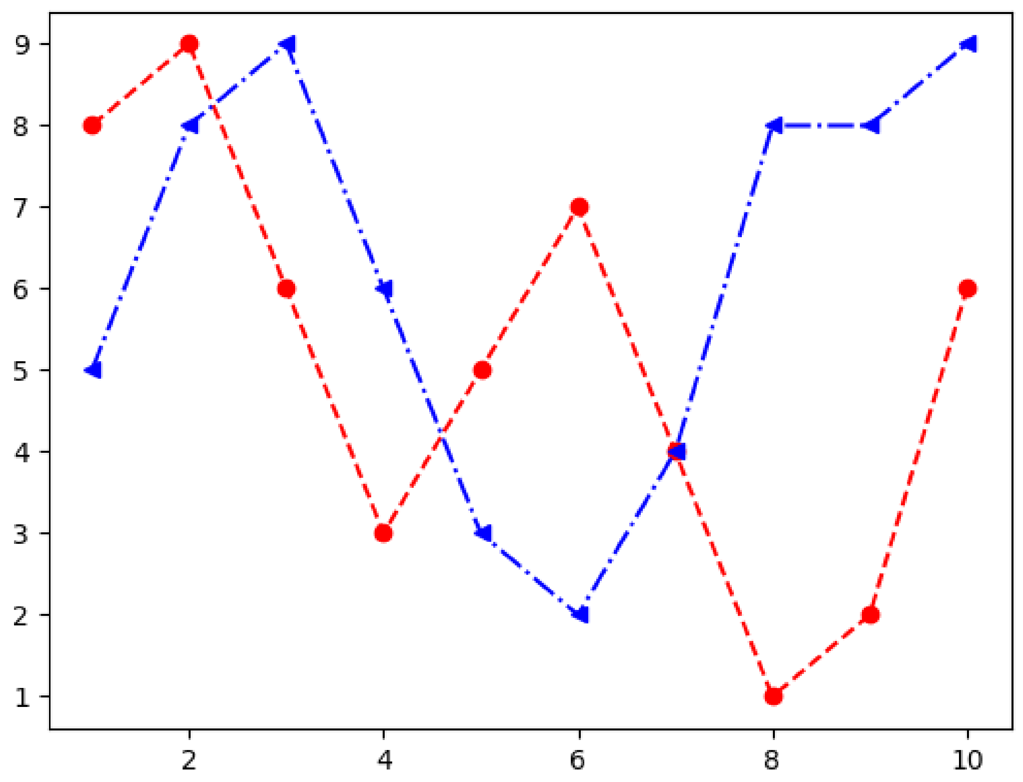
```
#import matplotlib below
import matplotlib.pyplot as plt

x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

# write a code to display the line chart of above x & y

plt.plot(x,y)
plt.show()
```



```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]

# write a code to display two lines in a line chart (data given above)
plt.plot(x,cxMarks)
plt.plot(x,cyMarks)
plt.show()
```

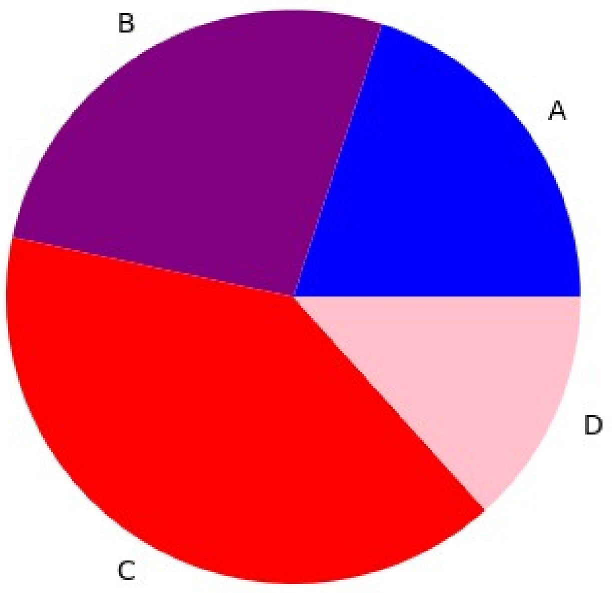


```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]


# write a code to generate below graph
plt.plot(x,cxMarks,marker='o',color='r',label='cxMarks',linestyle='--'
)
plt.plot(x,cyMarks,marker='<',color='b',label='cyMarks',linestyle='-.'
)
plt.show()
```



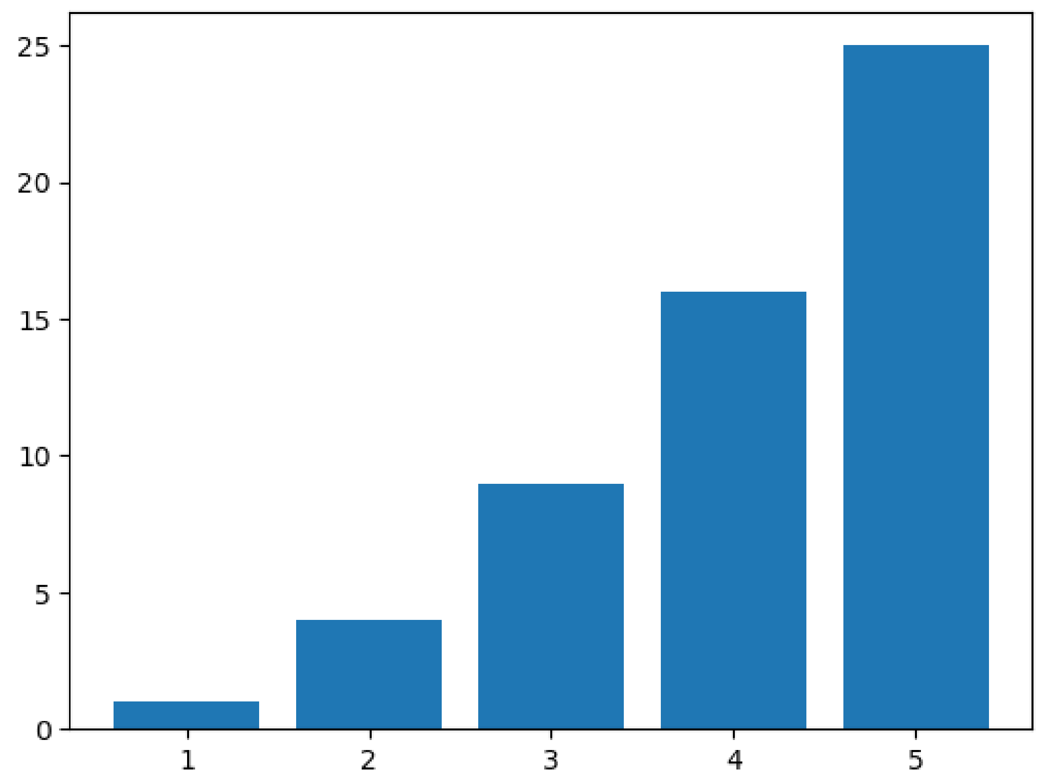
04) WAP to demonstrate the use of Pie chart.

```
#pie chart

sizes = [3,4,6,2]
labels = ['A','B','C','D']
colors = ['blue','purple','red','pink']
plt.pie(sizes,labels=labels,colors=colors)
plt.show()
```



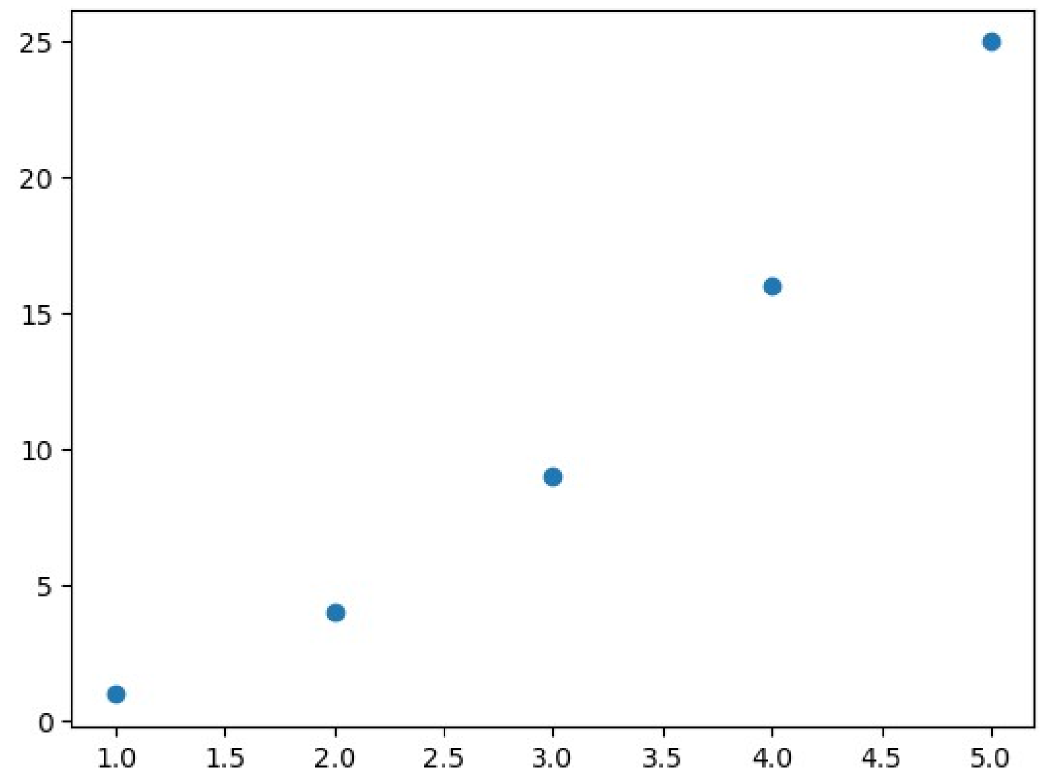
05) WAP to demonstrate the use of Bar chart.

```
x = [1,2,3,4,5]
y = [1,4,9,16,25]
plt.bar(x,y)
plt.show()
```



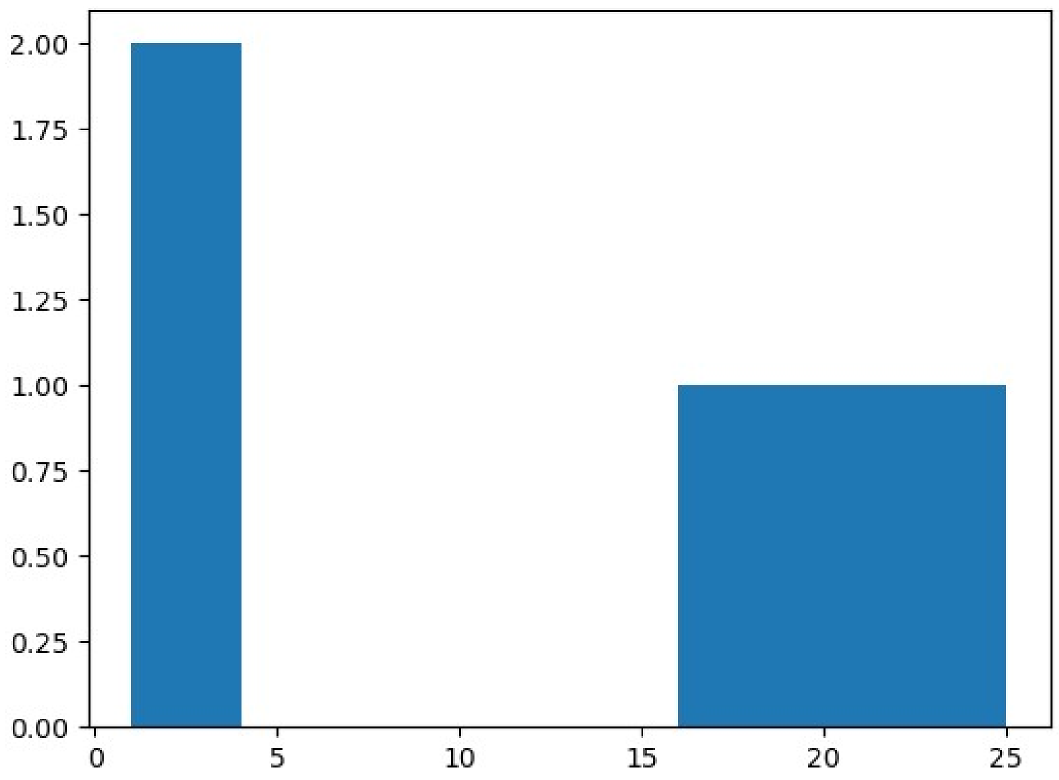
06) WAP to demonstrate the use of Scatter Plot.

```
x = [1,2,3,4,5]
y = [1,4,9,16,25]
plt.scatter(x,y)
plt.show()
```



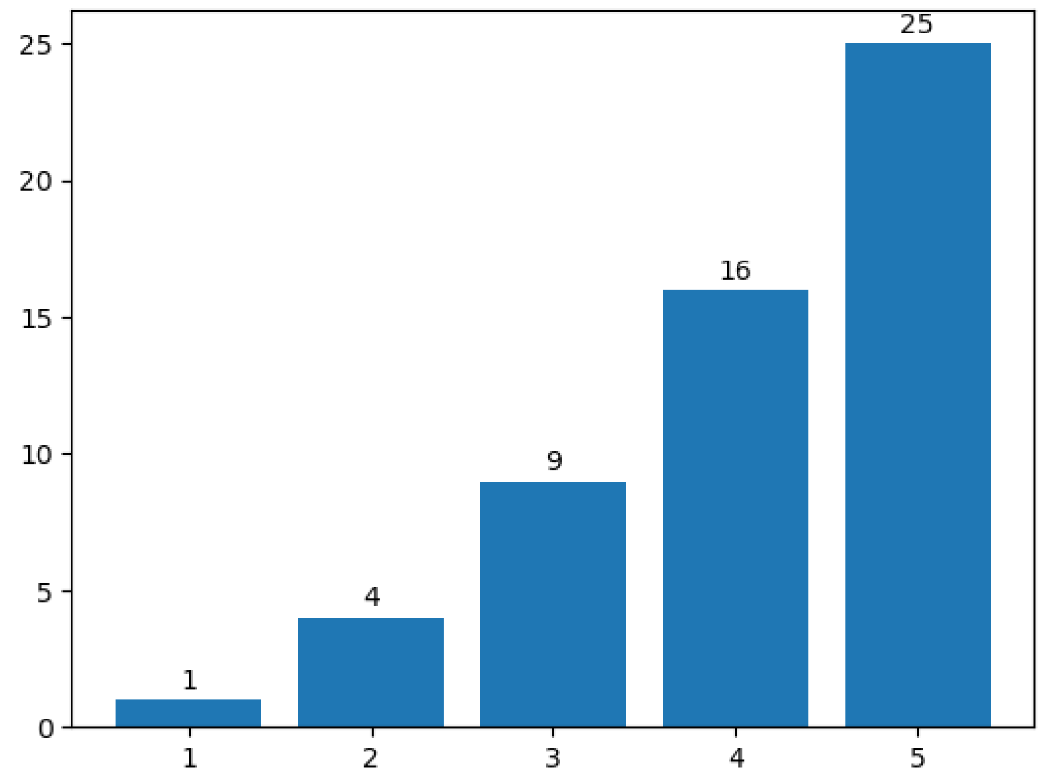
07) WAP to demonstrate the use of Histogram.

```
x = [1,20,3,40,50]
y = [1,4,9,16,25]

plt.hist(x,y)
plt.show()
```



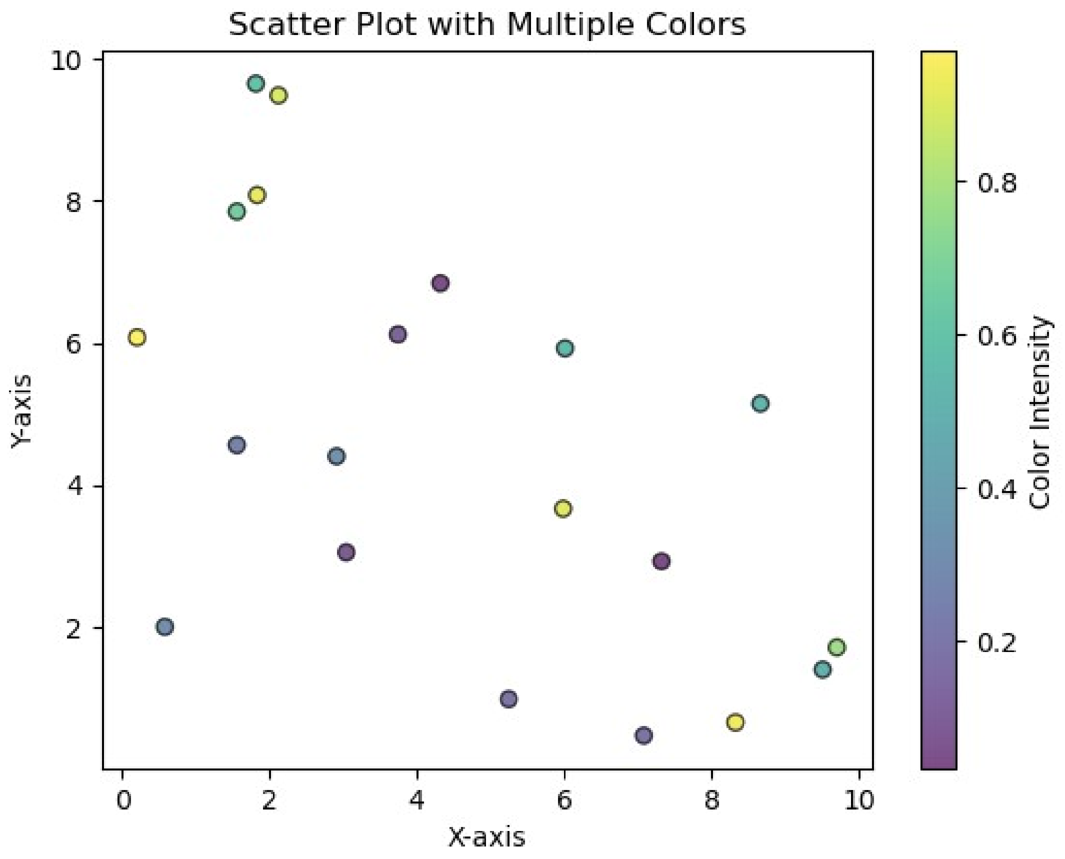
08) WAP to display the value of each bar in a bar chart using Matplotlib.

```
x = [1,2,3,4,5]
y = [1,4,9,16,25]
plt.bar(x,y)
for i, value in enumerate(y):
    plt.text(x[i], value + 0.2, str(value), ha='center', va='bottom')
plt.show()
```



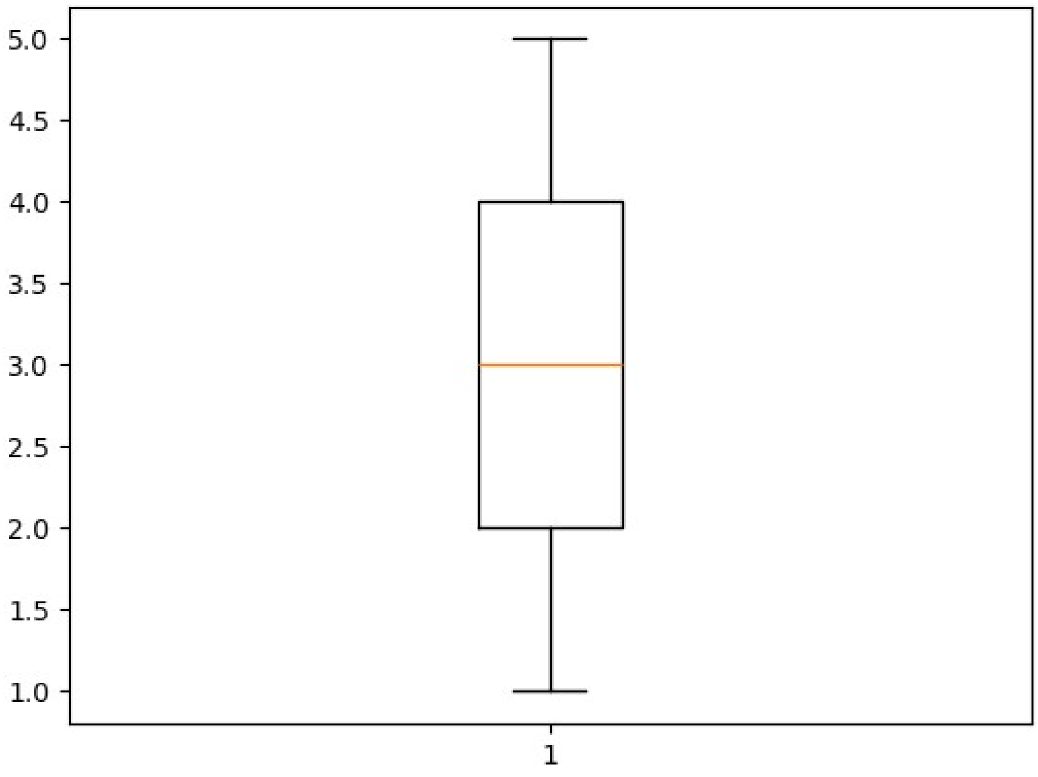
09) WAP create a Scatter Plot with several colors in Matplotlib?

```python
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(42)
x = np.random.rand(20) * 10
y = np.random.rand(20) * 10
colors = np.random.rand(20)

plt.scatter(x, y, c=colors, cmap='viridis', alpha=0.7,
edgecolors='black')


plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with Multiple Colors")

plt.colorbar(label="Color Intensity")

plt.show()
```



10) WAP to create a Box Plot.

```
x = [1,2,3,4,5]
plt.boxplot(x,horizontal=TRUE)
plt.show()
```


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1

OOP

01) Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

```
class Student:
    def __init__(self, name, age):
        print(f'Name: {name}, Age: {age}')

s1 = Student('MEHUL', 69)
```

```
Name: MEHUL, Age: 69
```

02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

```
class Bank_Account:

    def GetAccountDetails (self, acc_no, user_name, email, acc_type, 
balance):
        self.acc_no = acc_no
        self.user_name = user_name
        self.email = email
        self.acc_type = acc_type
        self.balance = balance

    def DisplayAccountDetails(self):
        print(self.acc_no)
        print(self.user_name)
        print(self.email)
        print(self.acc_type)
        print(self.balance)

a1 = Bank_Account()
a1.GetAccountDetails(1,2,3,4,5)
a1.DisplayAccountDetails()
```

```
1
2
3
```

```
4
5
```

03) WAP to create Circle class with area and perimeter function to find area and perimeter of circle.

```python
class circle:
    def __init__(self, r):
        self.r = r

    def calcPerimeter(self):
        print(2*3.14*self.r)

    def calcArea(self):
        print(3.14*(self.r ** 2))

c1 = circle(10)
c1.calcPerimeter()
c1.calcArea()
```

```
62.800000000000004
314.0
```

04) Create a class for employees that includes attributes such as name, age, salary, and methods to update and display employee information.

```python
class employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary

    def updateUser(self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary

    def displayUser(self):
        print(f'{self.name} {self.age} {self.salary}')

e1 = employee(1,2,3)
e1.updateUser(4,5,6)
e1.displayUser()
```

```
1 2 3
```

05) Create a bank account class with methods to deposit, withdraw, and check balance.

```
class bank:
    def __init__(self,cb):
        self.cb = cb

    def deposit(self, b):
        self.cb += b

    def withdraw(self, b):
        if self.cb < b:
            print("GARIBBBBB")
        else:
            self.cb -= b

    def display(self):
        print("Current Balance: ", self.cb)

b1 = bank(10000)
b1.deposit(10000)
b1.withdraw(5000)
b1.display()
```

```
Current Balance:  15000
```

06) Create a class for managing inventory that includes attributes such as item name, price, quantity, and methods to add, remove, and update items.

```
class inv:
    def add(self,name, price, qnt):
        self.name = name
        self.price = price
        self.qnt = qnt
        print(f'item added {self.name}')

    def update(self,name, price, qnt):
        self.name = name
        self.price = price
        self.qnt = qnt
        print(f'item updated {self.name}')

    def remove(self):
        self.name = ""
        self.price = ""
        self.qnt = ""
        print(f'item removed {self.name}')
```

```
c1 = inv()
c1.add(1,2,3)
c1.update(4,5,6)
c1.remove()
```

```
item added 1
item updated 4
item removed
```

07) Create a Class with instance attributes of your choice.

```
class MaroClass:
    def __init__ (self,name):
        self.name = name

m1 = MaroClass("bruce bhai")
m2 = MaroClass("meoweee")

print(m1.name)
print(m2.name)
```

```
bruce bhai
meoweee
```

08) Create one class student_kit

Within the student_kit class create one class attribute principal name (Mr ABC)

Create one attendance method and take input as number of days.

While creating student take input their name .

Create one certificate for each student by taking input of number of days present in class.

```
class student_kit:
    prn = "Mr. Pithadiya"
    def __init__(self, stn):
        self.stn = stn

    def atndnc(self, prdays):
        self.prdays = prdays

    def cirty(self):
        print(f'tamaro badak {self.stn} nishade {self.prdays} divas
hajar rahel chhe')

s1 = student_kit("yo")
s1.atndnc(123)
```

```
s1.cirty()
```

```
tamaro badak yo nishade 123 divas hajar rahel chhe
```

09) Define Time class with hour and minute as data member. Also define addition method to add two time objects.

```python
class samay:
    def __init__(self, hr, m):
        self.hr = hr
        self.m = m

    def addTimes(self, t1, t2):
        self.m = t1.m + t2.m
        self.hr = t1.hr + t2.hr
        if self.m >= 60:
            self.hr += 1
            self.m -= 60

        print(f'{self.hr}:{self.m}')

s1 = samay(0,0)
t1 = samay(1,60)
t2 = samay(2,30)
s1.addTimes(t1,t2)
```

```
4:30
```

Continued..

10) Calculate area of a ractangle using object as an argument to a method.

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def calculate_area(rect):
        return rect.length * rect.width


rect1 = Rectangle(10, 5)

area = calculate_area(rect1)

print(f"Area of Rectangle: {area}")
```

11) Calculate the area of a square.

Include a Constructor, a method to calculate area named area() and a method named output() that prints the output and is invoked by area().

```
class squre:
    def __init__(self, length):
        self.length = length

    def calculate_area(self,rect):
        area =  rect.length * rect.length
        self.output(area)

    def output(self,area):
        print(area)

sq = squre(5)
sq.calculate_area(sq)
```

```
25
```

12) Calculate the area of a rectangle.

Include a Constructor, a method to calculate area named area() and a method named output() that prints the output and is invoked by area().

Also define a class method that compares the two sides of reactangle. An object is instantiated only if the two sides are different; otherwise a message should be displayed : THIS IS SQUARE.

```python
class Rectangle:
    def __init__(self, length, width):
        if length == width:
            print("THIS IS SQUARE.")
        else:
            self.length = length
            self.width = width

    def area(self):
        area = self.length * self.width
        self.output(area)

    def output(self, area):
        print(f"Area of Rectangle: {area}")

    @classmethod
    def compare_sides(cls, length, width):
        if length == width:
            return False
        return True

length = float(input("Enter length: "))
width = float(input("Enter width: "))


if Rectangle.compare_sides(length, width):
    rect = Rectangle(length, width)
    rect.area()
else:
    print("Object not created because both sides are equal.")
```

```
Object not created because both sides are equal.
```

13) Define a class Square having a private attribute "side".

Implement get_side and set_side methods to accees the private attribute from outside of the class.

```
class squre:
    def __init__(self, side):
        self.__side = side

    def get_side(self):
        return self.__side

    def set_side(self, side):
        self.__side = side

s = squre(2)
s.get_side()
s.set_side(4)
```

```
2
```

14) Create a class Profit that has a method named getProfit that accepts profit from the user.

Create a class Loss that has a method named getLoss that accepts loss from the user.

Create a class BalanceSheet that inherits from both classes Profit and Loss and calculates the balanace. It has two methods getBalance() and printBalance().

```
class Profit:
    def getProfit(self):
        self.profit = float(input("Enter profit: "))

class Loss:
    def getLoss(self):
        self.loss = float(input("Enter loss: "))

class BalanceSheet(Profit, Loss):
    def getBalance(self):
        self.balance = self.profit - self.loss

    def printBalance(self):
        print(f"Balance: {self.balance}")

bs = BalanceSheet()
bs.getProfit()
bs.getLoss()
bs.getBalance()
bs.printBalance()
```

15) WAP to demonstrate all types of inheritance.

```
class Parent:
    def show(self):
        print("This is Parent class")

class Child(Parent):
    pass

# 2. Multiple Inheritance
class A:
    def displayA(self):
        print("Class A")

class B:
    def displayB(self):
        print("Class B")

class C(A, B):
    pass

# 3. Multilevel Inheritance
class Grandparent:
    def show_grandparent(self):
        print("This is Grandparent class")

class Parent(Grandparent):
    def show_parent(self):
        print("This is Parent class")

class Child(Parent):
    pass

# 4. Hierarchical Inheritance
class Base:
    def show_base(self):
        print("This is Base class")

class Derived1(Base):
    pass

class Derived2(Base):
    pass

# 5. Hybrid Inheritance
class X:
    def methodX(self):
        print("Class X")

class Y(X):
    def methodY(self):
        print("Class Y")

class Z(X):
    def methodZ(self):
        print("Class Z")

class W(Y, Z):
    pass

# Example Usage
o1 = Child()
o1.show()

o2 = C()
o2.displayA()
o2.displayB()

o3 = Child()
o3.show_grandparent()
o3.show_parent()

o4 = Derived1()
o4.show_base()

o5 = Derived2()
o5.show_base()

o6 = W()
o6.methodX()
o6.methodY()
o6.methodZ()
```

16) Create a Person class with a constructor that takes two arguments name and age.

Create a child class Employee that inherits from Person and adds a new attribute salary.

Override the **init** method in Employee to call the parent class's **init** method using the super() and then initialize the salary attribute.

```
class Person:
    def _init_(self, name, age):
        self.name = name
        self.age = age

    def display_info(self):
        print(f"Name: {self.name}, Age: {self.age}")

class Employee(Person):
    def _init_(self, name, age, salary):
        super()._init_(name, age)
        self.salary = salary

    def display_info(self):
        super().display_info()
        print(f"Salary: {self.salary}")

emp = Employee("John wick", 30, 50000)
emp.display_info()
```

17) Create a Shape class with a draw method that is not implemented.

Create three child classes Rectangle, Circle, and Triangle that implement the draw method with their respective drawing behaviors.

Create a list of Shape objects that includes one instance of each child class, and then iterate through the list and call the draw method on each object.

```
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def draw(self):
        pass

class Rectangle(Shape):
    def draw(self):
        print("Drawing a rectangle")

class Circle(Shape):
    def draw(self):
        print("Drawing a circle")

class Triangle(Shape):
    def draw(self):
        print("Drawing a triangle")

shapes = [Rectangle(), Circle(), Triangle()]

for shape in shapes:
    shape.draw()
```

```
Drawing a rectangle
Drawing a circle
Drawing a triangle



---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 8
      5         print(color)
      7 antt = ant()
----> 8 antt.color()

TypeError: ant.color() missing 1 required positional argument: 'color'
```